OGILVY
RENAULT
LLP / S.E.N.C.R.L., s.r.l.

October 28, 2005

Mail Stop PGPUB

RECEIVED
USPTO-PG PUBS
OCT 28 2005

Commissioner for Patents
United States Patent and Trademark Office
P.O. Box 1450
Alexandria, VA 22313-1450
U.S.A.

Dear Sir:

ERRORS NOTED IN APPLICATION FOLLOWING PUBLICATION

RE: United States Patent Application Serial No. 10/787,107
Title: METHOD AND APPARATUS FOR DERIVING OPTIMAL PATHS THROUGH A NETWORK SUBJECT TO A SUBSET SEQUENCE CONSTRAINT
Inventors: Hong ZHANG et al.
Assignee: Nortel Networks Limited
Our File: 9-13528-202US

A review of the above-identified patent application following publication has identified the following errors in the application:

In the published application, the following paragraphs should be amended to show a format of the identifiers as they appeared in the application as filed:

[0034] FIG. 2 is a schematic diagram of an undirected graph 18 that represents the network shown in FIG. 1. The graph 18 is used for computing paths through the network 8. All of the NEs 10/SNEs 11 and links 12 that are represented (by corresponding nodes 20, and edges 22) have capacity available to transport traffic at a predefined rate. Each NE 10/SNE 11 is represented by a node 20 (or vertex), and each link 12 is represented by a corresponding edge 22 (only 3 of which are labeled). In the description that follows, particular nodes are identified by their labeled letter (e.g. node **c**, node **d**), and edges are identified by the labeled letters of the two nodes connected to the edge in either order (e.g. edge **ge**). Likewise a path through the graph 18 is a sequence of labeled letters where each adjacent pair of letters corresponds to an edge of the graph. An adjacent node to a node **n** is a node **m** for which an edge **nm** exists.

[0036] Several examples of subset sequence constraints are encountered on data transport networks having a subnetwork 14 as described above. A first constraint is called subset intransitivity. The restriction imposed by subset intransitivity is that no path may include two consecutive edges between three nodes in the serial restriction group. To make this limitation more clear, consider the graph 18 shown in FIG. 2, the serial restriction group **b,c,e,f** contains 6 edges 22: **bc**, **be**, **bf**, **ce**, **cf**, and **ef**. According to the subset intransitivity, the paths **abce**, **bcf**, **bcbadc** are not allowable; whereas **bace**, **abcda**, **bcdce**, and **bcadce** are allowable. It will be noted that paths that fail this rule conspicuously include three consecutive members of the serial restriction group.

[0038] There are stronger restrictions that may also be imposed on allowable routes in data transport networks. For example, in accordance with another, stronger, subset sequence constraint, no two adjacent subnetwork links may be a part of the path, in sequence, though not necessarily consecutively. In other words, if **x,y,z** are in the same subnetwork, a path that contains **xy** and then later **yz**, is not allowable. Because optional routing naturally precludes self intersecting (looping) paths, the stronger constraint is effectively equally well expressed as follows: no path can contain **xy** and **yz**, because any path containing **yz..xy** is looped. According to this subset sequence constraint the following are not allowable paths: **abce**, **bcdce**, **ecdfbacf**, **bcadce**; whereas **abcdf**, **ecdfb**, **cbacf** (although this path may not be chosen by any optional routing algorithm because of the loop at node **c**) are allowable.

[0041] FIG. 4 schematically illustrates a graph 19 representing the network illustrated in FIG. 3, with edge weights likewise assigned. The graph 19 includes nodes **b,c,d,e** in a first serial restriction group associated with the first subnetwork 14a, and nodes **g,h,i,j**, which are in a second serial restriction group associated with the second subnetwork 14b.

[0053] Two examples of the method described above are presented. Assuming the root node is node **a** of FIG. 2, the procedure begins by assigning a weight of 0 to label L_1(**a**), and a path of {**a**}. The node **a** is then permanently labeled and label L_1(**a**) is removed from the priority queue. After the procedure completes processing of **a**, the adjacent nodes are respectively labeled; i.e., L_1(**b**)=(1,{**ab**}), L_1(**c**)=(5,{**ac**}), and L_1(**d**)=(4,{**ad**}). The processing of **a** is now complete, and of the labels in the priority queue, label L_1(**b**) has a lowest cost (1).

OGILVY
RENAULT
LLP/S.E.N.C.R.L., s.r.l.

[0054] Accordingly, the label L_1(b) is made permanent and processed next. As node b is in the serial restriction group, it has a backup label that happens to be initialized. Because the restriction flag is not set at node b, the backup label is removed from the priority queue. The labels of nodes adjacent to node b include c,e,f, and the already permanent node a. Accordingly, c,e,f, are labeled as L_1(c)=(2,{abc}), L_1(e)=(6,{abe}), and L_1(f)= (5,{abf}), respectively. Because node b is in the serial restriction group, and node c, and node e are also in the serial restriction group, the former primary labels of node c and node e are saved as backup labels. Accordingly, label L_2(c)=(5,{ac}), and the initialized values of label L_1(e) and label L_1(f) are stored as the label L_2(e) and label L_2(f), respectively. Further the restriction flags are set at each of nodes c,e,f, to preclude the extension of the primary labels to another member of the serial restriction group.

[0055] The procedure iterates making node c permanent but because the restriction flag is set at node c, the backup label for node c remains temporary, permitting the procedure to return to node c, if and when the backup label identifies a least cost path called, herein a secondary path. Processing the primary label of node c involves updating labels of nodes d,e,f. The primary label of node d is updated as the cost of the path through node c is 3, which is less than the existing path ad with a cost of 4, i.e. label L_1(d)=(3,{abcd}). The path abcd is allowable because node d is not a member of the serial restriction group. Paths abce and abcf are not allowable by the subset intransitivity (and by the stronger subset sequence constraint) and accordingly neither of these labels are changed. Specifically, because the serial restriction flag set at node c, node e and node f are not updated.

[0056] In a next iteration, the label at node d is made permanent. The nodes c and f are the only nodes adjacent to node d that have labels in the priority queue. The backup label of node c, and the primary label of node f may be subsequently updated. The primary label of node f is changed to the label L_1(f)=(4,{abcdf}). Because node d is not in the serial restriction group, the primary label path is now extendible to a node in the group. Accordingly, the restriction flag at node f is unset, and the backup label at node f is reinitialized. The procedure will not update label L_2(c) with the path abcdc if the subset sequence constraint is the stronger constraint, but will update label L_2(c) if the subset sequence constraint is subset intransitivity. In this example, the subset sequence constraint is the stronger constraint, and accordingly the backup label of node c is not updated, specifically because node c is in the path of the label of node d.

[0057] It will be noted that if the subset sequence constraint is subset intransitivity, the resulting labels (in the order in which they are processed) are: L_1(a)=(0,{a}), L_1(b)=(1,{ab}), L_1(c)= (2,{abc}), L_1(d)=(3, {abcd}), L_2(c)=(4,{abcdc}), L_1(f)= (4,{abcdf}), L_1(e)=(5, {abcdce}), and L_1(g)=(6,{abcdfg}). When the procedure ends a backup label at node e (i.e. L_2(e)= (7,{abcdfge})) remains in the priority queue.

[0058] The labels in the priority queue are now as follows: L_1(f)=(4,{abcdf}), L_2(c)=(5,{abc}), L_1(e)= (6,{abe}), and initialized labels L_2(e), L_2(f), L_1(g), and L_2(g). Consequently the next label chosen is at node f. The backup label of node f is dropped because the restriction flag at node f is unset, and the label L_1(f) is made permanent. The adjacencies of node f with impermanent labels are node e and node g. Accordingly the corresponding labels are updated as follows: $_1$(e)=(5,{abcdfe}), and L_1(g)=(6,{abcdfg}). The restriction flag remains set at node e, and the (initialized) backup label of node e is unchanged. No restriction flag is defined for node g as node g is not in the serial restriction group.

[0059] The label L_2(c) is next made permanent, but the only impermanent label of an adjacent node (e) already has a lower weight (5), and accordingly is not changed. Next the primary label of node e is made permanent but as the restriction flag is set at node e, label L_2(e) is retained in the priority queue. The only label in the priority queue of a node adjacent to node e, is node g. When label L_1(g) is updated, no change takes place because of an equal cost. Finally the label of node g is made permanent, label L_2(e) is updated to label L_2(e)=(7,{abcdfge}) and the procedure ends.

[0061] The second network 9 provides another exemplary graph 19 to which the illustrated method may be applied. The second graph 19 has two serial restriction groups, each associated with a respective subset sequence constraint. It will be noted that the two serial restriction groups are disjoint, and independently preclude allowance of respective paths. Starting again with a node a, the primary label of node a is made permanent, and the primary labels L_1(b)=(1,{ab}) and L_1(c)=(3,{ac}) are updated. Subsequently, label L_1(b) is made permanent, its backup label is dropped, and label L_1(d) is set to (3,{abd}). The restriction flag is set at node d, and its backup label is effectively reinitialized. The primary label at node c is not of a higher cost than a path through node b extended to node c via bc, and so label L_1(c) is not changed. In a next iteration, the label at node c is made permanent, its backup label is removed from the priority queue, and primary labels of node f and node e are changed to L_1(f)=(5,{acf}), and L_1(e)=(4,{ace}), respectively. The restriction flag is set at node e. Neither label L_1(d) nor label

L_2(d) is changed when node c is processed because node d has a lower cost primary label than that of the path through node c, and because the path through node c is not allowably extended, the secondary path to node d cannot be updated. In a next iteration the primary label at node d is made permanent, its initialized backup label is retained, and no adjacent label is updated because the restriction flag is set at node d., and the only impermanent label of an adjacent node is at node e, which is in the same serial restriction group as node d.

[0062] A next iteration makes the label at node e permanent. The backup label at node e is retained in the priority queue. The backup label at node d is not updated because aced is not allowable and the restriction flag is set at node e, and node f is not relabeled by the update because it already has a lower cost route than an extension of the path through node e. However, the primary label of the path to node g is set to (5,{aceg}). While the subset restriction flag is set at node e, and node g is in a serial restriction group, the path is still permitted because node e and node g are in different serial restriction groups (corresponding to subnetworks 14a and 14b, respectively). The primary label of node f is the next label made permanent, resulting in node h being updated, which receives a label L_1(h)=(7,{acfh}). The backup label of node e is also updated and changed from the initialized value to (8,[acfe]).

[0063] In a subsequent iteration, the primary label of node g is made permanent, its backup label is dropped, and node i, and node j are relabeled as follows: L_1(i)=(8,{acegi}) L_1(j)=(9,{acegj}), and restriction flags are set at both of these nodes. Node h is also updated so that the restriction flag is set, and the labels are changed as follows: L_1(h)=(6,{acegh}), and L_2(h)=(7,{acfh}). Depending on whether the subset sequence constraint is the stronger constraint or subset intransitivity, label L_2(e) is not updated, or is updated to (6,{acege}). In this example, the network is subject to the stronger subset sequence constraint.

[0064] At this juncture, the priority queue contains the following labels: L_1(h)=(6,{acegh}), L_2(h)=(7,{acfh}), L_2(e)=(8,{acfe} [alternatively (6,{acege}), if merely subset intransitive], L_1(i)=(8,{acegi}), L_1(j)=(9,{acegj}) and backup labels of nodes d, i and j are initialized. In a next iteration, the primary label of node h is therefore made permanent, but no labels can be updated as all impermanently labeled nodes adjacent to node h are in the same serial restriction group as node h, and the restriction flag is set at node h. The backup label of node h is subsequently made permanent, and node h is processed again via a secondary path acfh. Subsequently the primary label of node j is updated as follows label

OGILVY RENAULT
LLP / S.E.N.C.R.L., s.r.l.

L_1(j)=(8,{acfhj}), and the restriction flag remains set at node j because node h and node j are in the same serial restriction group. The primary label of node i is also updated but results in no change.

[0065] The backup label of node e is next to be made permanent (if the subset sequence constraint were subset intransitivity, the order in which labels L_1(h), L_2(h)and L_2(e) are processed is different, but nothing else changes viz. primary labels). Label L_2(d) cannot be updated as the path acfed (or equally aceged) is not allowably extended to a node in the serial restriction group. The primary label of node i is next made permanent: its initialized backup label is retained, and again no other label can be updated because of the restriction flag set at node i. Lastly node j is made permanent, its still initialized backup label is also retained and the priority queue contains only backup labels for nodes d, i, and j. The procedure therefore ends.

In addition, the following amendments should also be made:
In paragraph [0063], line 4 "{eg}" should read --{acegj}--; line 6 "{aceh}" should read --{acegh}--. In paragraph [0064], line 2 "{aceh)})" should read --{acegh})--; line 4, "L_1(i)=(9" should read –L_1(j)=(9 and "L_1(i)=(8, {acf j})" should read --L_1(j)=(8,{acfhj})--

To facilitate in understanding the amendments requested, a copy of the published application is enclosed showing the errors.

While republication of the application is not required, the Office is hereby requested to correct this error in the Office records, and confirmation that this error has been corrected is requested.

Yours very truly,

[illegible]

Max R. Wood
Reg. No. 40,388
Agent of Record

MRW/ma

optimal among the paths to nodes outside of the tree; and if the path to the node cannot be extended to a node in a serial restriction group that includes some of the nodes in the graph, re-including the node through a secondary path to the node that can be extended to a node in the serial restriction group, when the secondary path is most optimal among the paths to nodes outside of the tree.

BRIEF DESCRIPTION OF THE DRAWINGS

[0023] Further features and advantages of the present invention will become apparent from the following detailed description, taken in combination with the appended drawings, in which:

[0024] **FIG. 1** is a schematic diagram representing a mesh connected network that includes a subnetwork;

[0025] **FIG. 2** is a schematic diagram illustrating a weighted graph representing the network shown in **FIG. 1**;

[0026] **FIG. 3** is a schematic representing a second mesh connected network that includes two subnetworks;

[0027] **FIG. 4** is a schematic diagram illustrating a weighted graph representing the network shown in **FIG. 3**; and

[0028] **FIG. 5** is a flow chart illustrating principal steps involved in computing least cost paths through a weighted graph.

[0029] It should be noted that throughout the appended drawings, like features are identified by like reference numerals.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENT

[0030] The invention provides a method of deriving an optimal allowable route through a network of network elements (NEs) interconnected by links, where the allowable route through the network are subject to a subset sequence constraint that forbids certain sequences of NEs (or their links) with respect to membership of the NEs in a serial restriction group. The method permits optimized routing across networks that have intransitive serial restriction groups, such as networks having abstracted subnetworks, and other restrictions.

[0031] **FIG. 1** schematically illustrates a network **8** of network elements **10** (NEs), including subnetwork elements SNEs **11** interconnected by links **12**. Respective NEs **10** and SNEs **11** are identified as NEa, SNEb, . . . NEg. The network **8** includes a subnetwork **14** encompassing SNEb, SNEc, SNEe, and SNEf, and six links **12** interconnecting these four, which have of darker line weight for easy identification. It will be recognized by those of skill in the art that the subnetwork is a completely interconnected (i.e. full mesh) subnetwork.

[0032] As stated above, routing constraints and continuity conditions of subnetwork **14** impose limitations on allowable paths that pass through the subnetwork **14**, and, in particular, introduce a subset sequence constraint (i.e. a limitation on allowable paths through a network that depends on a sequence of SNEs **11** in the subnetwork **14**). The subnetwork **14** may not correspond directly to the topology of an underlying physical subnetwork, for various reasons. A method for computing metric information on such subnetworks is described in co-pending, co-assigned U.S. patent application Ser. No. 10/718,681 entitled METHOD AND APPARATUS FOR COMPUTING METRIC INFORMATION FOR ABSTRACTED NETWORK LINKS, filed on Nov. 24, 2003 which is incorporated herein by reference.

[0033] A route selection processor **16** is shown residing in a NEa **10** of the network **8**. The route selection processor **16** includes hardware and software for executing a process in accordance with the invention. The route selection processor **16** could also be located at any other location in the network **8**, or a network management system known in the art.

[0034] **FIG. 2** is a schematic diagram of an undirected graph **18** that represents the network shown in **FIG. 1**. The graph **18** is used for computing paths through the network **8**. All of the NEs **10**/SNEs **11** and links **12** that are represented (by corresponding nodes **20**, and edges **22**) have capacity available to transport traffic at a predefined rate. Each NE **10**/SNE **11** is represented by a node **20** (or vertex), and each link **12** is represented by a corresponding edge **22** (only 3 of which are labeled). In the description that follows, particular nodes are identified by their labeled letter (e.g. node c, node d), and edges are identified by the labeled letters of the two nodes connected to the edge in either order (e.g. edge gc). Likewise a path through the graph **18** is a sequence of labeled letters where each adjacent pair of letters corresponds to an edge of the graph. An adjacent node to a node n is a node m for which an edge nm exists.

[0035] The nodes **20** that represent NEs of the subnetwork **14**, are underlined to indicate membership in a serial restriction group. The serial restriction group is defined as a subset of the nodes **20** (i.e. {b,c,e,f}) that are used to identify paths that are not allowable because of a subset constraint.

[0036] Several examples of subset sequence constraints are encountered on data transport networks having a subnetwork **14** as described above. A first constraint is called subset intransitivity. The restriction imposed by subset intransitivity is that no path may include two consecutive edges between three nodes in the serial restriction group. To make this limitation more clear, consider the graph **18** shown in **FIG. 2**, the serial restriction group b,c,e,f contains 6 edges **22**: bc, be, bf, ce, cf, and ef. According to the subset intransitivity, the paths abce, bcf, bcbade are not allowable; whereas bace, abcda, bcdce, and bcadce are allowable. It will be noted that paths that fail this rule conspicuously include three consecutive members of the serial restriction group.

[0037] As used herein, a first and a second node are "sequential" or "in sequence" in a path if the first appears before the second. Non-adjacent nodes in a path may be said to be sequential with respect to an order of the nodes. Similarly edges ab, and bc are said to be sequential in a path if ab and bc are included in the path, and ab appears (not necessarily directly) before bc.

[0038] There are stronger restrictions that may also be imposed on allowable routes in data transport networks. For example, in accordance with another, stronger, subset sequence constraint, no two adjacent subnetwork links may be a part of the path, in sequence, though not necessarily consecutively. In other words, if x,y,z are in the same

BEST AVAILABLE COPY

subnetwork, a path that contains xy and then later yz, is not allowable. Because optional routing naturally precludes self intersecting (looping) paths, the stronger constraint is effectively equally well expressed as follows: no path can contain xy and yz because any path containing yz . . . xy is looped. According to this subset sequence constraint the following are not allowable paths: abce, bcdce, ecdfbacb, bcadce whereas abcdf, ecdfb, cbacf (although this path may not be chosen by any optional routing algorithm because of the loop at node c) are allowable.

[0039] A still stronger subset sequence constraint that can be accommodated using a variant of the method described below, is a restriction that no two links between any members of a serial restriction group can be included in a path. In other words, no path may include three consecutive nodes in the serial restriction group, or any two pair of consecutive nodes in the serial restriction group. Other restrictions can also be enforced by a rule or an explicit list of allowable paths, if required. A clear rule for determining which paths are allowed, and which are allowably extended to another node in the serial restriction group, are required by the procedure described below, and in order to ensure optimality, the process requires that the subset sequence constraint is monotonic (any extension of a path through the graph is not allowable, if the path itself is not allowable), and that allowability is not dependent on more than a two nodes in the path. Of course the invention can be adapted to handle more intricate constraints.

[0040] FIG. 3 schematically illustrates a second network 9 for use in illustrating the invention. Network 9 includes two subnetworks 14*a*,*b*, a first of which (14*a*) includes NEb, NEc, NEd, and NEe, and five links 12 interconnecting these four to each other, and a second subnetwork 14*b* including NEg, NEh, NEi, and NEj, and the corresponding six links 12. The first subnetwork 14*a* is full mesh connected save one link between NEb and NEe, and the second subnetwork 14*b* is full mesh connected. The elements of the subnetworks 14*a*,*b* are shown with a darker line weight for easy identification.

[0041] FIG. 4 schematically illustrates a graph 19 representing the network illustrated in FIG. 3, with edge weights likewise assigned. The graph 19 includes nodes b,c,d,e in a first serial restriction group associated with the first subnetwork 14*a*, and nodes g,h,i,j which are in a second serial restriction group associated with the second subnetwork 14*b*.

[0042] The invention provides a method for determining an optimal route through a network that is allowable in accordance with any subset sequence-constraint in the path. The method is generally modeled after the well-known Dijkstra's algorithm, which identifies least-cost paths to all of the nodes from a given root node for a positively weighted graph. Effectively, Dijkstra's algorithm constructs a tree rooted at the root node that spans a weighted graph. The tree is constructed by iteratively expanding the tree to include a node for which a path from the root to the node is optimal among the paths to nodes not in the tree. Optimality is determined with respect to an optimization parameter computed using weights of the edges of the graph. The optimization parameter is typically computed as a sum of the (additive) weights of the edges in the path, although it may be a minimum or maximum of a (convex) weight of the edges in the path, or may be computed otherwise. Hereinafter the optimization parameter will be assumed to be a "cost", wherein the weights are additive and lower costs are more optimal. The weight of the edges is associated with metric information relating to resource availability in the network.

[0043] Procedurally, Dijkstra's algorithm starts with a permanently labeled root node (which is assigned a path containing only an identifier of the root node, and a null cost), and for every node adjacent to the root node, a weight of an edge connecting the root node to the adjacent node is used to assign a temporary label to the adjacent node. The label includes the path of the root node extended to the adjacent node, and a cost associated with the label is a sum of the weight of the edge and a cost of the path of the root node. The temporary label with the least cost is then selected and made permanent. All of the temporarily labeled nodes adjacent to the node that is permanently labeled are updated to ensure that the path to the temporary labeled node is optimal. That is, if a sum of the cost associated with the permanently labeled node and the weight of an edge between the permanently labeled node and an adjacent node has a lower cost than a cost associated with the adjacent node's current temporary label, a lower cost path between the root node and the node adjacent to the most recently permanently labeled node is available. Accordingly, the current temporary label of the adjacent node is replaced with a label identifying a path that is equal in cost to the path identified by the permanent label with the adjacent node appended thereto. The procedure is then reiterated to select and make permanent a temporary label having a least cost. Once all of the nodes are permanently labeled, least cost paths are identified by each label. Making the labels permanent is the way in which the tree is iteratively expanded to include nodes in the graph, until all of the nodes are permanently labeled.

[0044] In accordance with the invention, the subset sequence constraints on allowable paths are compensated for by precluding the temporary labeling of paths that are not allowable, and adding backup labels for each node that may potentially have different primary and secondary optimal paths. A primary path is a least cost path to the node. A secondary path is a least cost path to the node that can be extended to any other adjacent node, given a subset sequence constraint. In accordance with the present embodiment, a serial restriction group is defined to facilitate identification of allowable paths and paths that are allowed to be extended. Additionally, a restriction flag for each serial restriction group can be set at any node in the serial restriction group, to expedite the process. Nodes that are not in a serial restriction group, do not require backup labels or restriction flags. The method provides the backup labels so that if a least cost path to a node in the serial restriction group is allowable, but cannot be extended to another node in the serial restriction group, a backup label is stored that identifies a higher cost path that can be extended to a node in the serial restriction group. The backup labels permit the re-inclusion of nodes in the tree via a secondary path that can be extended to another node in the serial restriction group.

[0045] The method described with reference to FIG. 5 compensates for the subset sequence constraint on allowable cost, and this may be determined prior to the comparison in step 86. If the new path is of a lower cost, the restriction flag at node m is unset, the label $L_2(m)$ is reinitialized (step 84), and the procedure advances to step 80, where the label $L_1(m)$ is replaced with a label that identifies a lower cost new path through node n. If the new path is not less costly than the one identified by the label $L_1(m)$, it is determined (step 86) whether the new path is less costly than the path identified by the label $L_2(m)$. If the new path is found to be at least as costly as the one identified by the label $L_2(m)$, the procedure returns to step 60 without updating any label of node m. On the other hand, if the new path is less costly than a current secondary path to node m, (the path identified by the label $L_2(m)$), one of two procedures for changing the label $L_2(m)$ is applied, in dependence on whether the subset sequence constraint is subset intransitivity or a stronger constraint. If the process only compensates for subset intransitivity, the procedure advances to step 88 wherein the label $L_2(m)$ is replaced with a label identifying the new path, and having the corresponding lower cost, before the procedure returns to step 60. If the process compensates for the stronger subset sequence constraint, the procedure advances to step 87 in which it is determined whether the path identified by the label $L(n)$ contains node m. If node m is in the path to node n, the procedure precludes the updating of the secondary path to node m, and in this way avoids a looping path. If node m is determined not to be in the path to node n, the procedure advances to step 88.

[0053] Two examples of the method described above are presented. Assuming the root node is node a of FIG. 2, the procedure begins by assigning a weight of 0 to label $L_1(a)$, and a path of {a}. The node a is then permanently labeled and label $L_1(a)$ is removed from the priority queue. After the procedure completes processing of a, the adjacent nodes are respectively labeled; i.e. $L_1(b)=(1, \{ab\})$, $L_1(c)=(5, \{ac\})$, and $L_1(d)=(4, \{ad\})$. The processing of a is now complete, and of the labels in the priority queue, label $L_1(b)$ has a lowest cost (1).

[0054] Accordingly, the label $L_1(b)$ is made permanent and processed next. As node b is in the serial restriction group, it has a backup label that happens to be initialized. Because the restriction flag is not set at node b, the backup label is removed from the priority queue. The labels of nodes adjacent to node b include c,e,f, and the already permanent node a. Accordingly, c,e,f are labeled as $L_1(c)=(2, \{abc\})$, $L_1(e)=(6, \{abe\})$, and $L_1(f)=(5, \{abf\})$, respectively. Because node b is in the serial restriction group, and node c, and node e are also in the serial restriction group, the former primary labels of node c and node e are saved as backup labels. Accordingly, label $L_2(c)=(5; \{ac\})$, and the initialized values of label $L_1(e)$ and label $L_1(f)$ are stored as the label $L_2(e)$ and label $L_2(f)$ respectively. Further the restriction flags are set at each of nodes c,e,f, to preclude the extension of the primary labels to another member of the serial restriction group.

[0055] The procedure iterates making node c permanent but because the restriction flag is set at node c, the backup label for node c remains temporary, permitting the procedure to return to node c, if and when the backup label identifies a least cost path called, herein a secondary path. Processing the primary label of node c involves updating labels of nodes d,e,f. The primary label of node d is updated as the cost of the path through node c is 3, which is less than the existing path ad with a cost of 4, i.e. label $L_1(d)=(3, \{abcd\})$. The path abcd is allowable because node d is not a member of the serial restriction group. Paths abce and abcf are not allowable by the subset intransitivity (and by the stronger subset sequence constraint) and accordingly neither of these labels are changed. Specifically, because the serial restriction flag set at node c, node e and node f are not updated.

[0056] In a next iteration, the label at node d is made permanent. The nodes c and f are the only nodes adjacent to node d that have labels in the priority queue. The backup label of node c and the primary label of node f may be subsequently updated. The primary label of node f is changed to the label $L_1(f)=(4, \{abcdf\})$. Because node d is not in the serial restriction group, the primary label path is now extendible to a node in the group. Accordingly, the restriction flag at node f is unset, and the backup label at node f is reinitialized. The procedure will not update label $L_2(c)$ with the path abcdc if the subset sequence constraint is the stronger constraint, but will update label $L_2(c)$ if the subset sequence constraint is subset intransitivity. In this example, the subset sequence constraint is the stronger constraint, and accordingly the backup label of node c is not updated, specifically because node c is in the path of the label of node d.

[0057] It will be noted that if the subset sequence constraint is subset intransitivity, the resulting labels (in the order in which they are processed) are: $L_1(a)=(0, \{a\})$, $L_1(b)=(1, \{ab\})$, $L_1(c)=(2, \{abc\})$, $L_1(d)=(3, \{abcd\})$, $L_2(c)=(4, \{abcdc\})$, $L_1(f)=(4, \{abcdf\})$, $L_1(e)=(5, \{abcdce\})$ and $L_1(g)=(6, \{abcdfg\})$. When the procedure ends a backup label at node e (i.e. $L_2(e)=(7, \{abcdfge\})$) remains in the priority queue.

[0058] The labels in the priority queue are now as follows: $L_1(f)=(4, \{abcdf\})$, $L_2(c)=(5, \{abc\})$, $L_1(e)=(6, \{abe\})$, and initialized labels $L_2(e)$, $L_2(f)$, $L_1(g)$, and $L_1(g)$. Consequently the next label chosen is at node f. The backup label of node f is dropped because the restriction flag at node f is unset, and the label $L_1(f)$ is made permanent. The adjacencies of node f with impermanent labels are node e and node g. Accordingly the corresponding labels are updated as follows: $L_1(e)=(5, \{abcdfe\})$ and $L_1(g)=(6, \{abcdfg\})$. The restriction flag remains set at node e and the (initialized) backup label of node e is unchanged. No restriction flag is defined for node g as node g is not in the serial restriction group.

[0059] The label $L_2(c)$ is next made permanent, but the only impermanent label of an adjacent node (e) already has a lower weight (5), and accordingly is not changed. Next the primary label of node e is made permanent but as the restriction flag is set at node e, label $L_2(e)$ is retained in the priority queue. The only label in the priority queue of a node adjacent to node e is node g. When label $L_1(g)$ is updated, no change takes place because of an equal cost. Finally the label of node g is made permanent, label $L_2(e)$ is updated to label $L_2(e)=(7; \{abcdfge\})$ and the procedure ends.

[0060] It should be noted that the procedure can end at an earlier time. Specifically when a path (corresponding to a route) between a predefined set of nodes (representing respective NEs) has been obtained, the procedure may end. An optimal path to a node is designated when the label is made permanent.

[0061] The second network 9 provides another exemplary graph 19 to which the illustrated method may be applied.

The second graph 19 has two serial restriction groups, each associated with a respective subset sequence constraint. It will be noted that the two serial restriction groups are disjoint, and independently preclude allowance of respective paths. Starting again with a node a, the primary label of node a is made permanent, and the primary labels $L_1(b)=(1, \{ab\})$ and $L_1(c)=(3, \{ac\})$ are updated. Subsequently, label $L_1(b)$ is made permanent, its backup label is dropped, and label $L_1(d)$ is set to (3, {abd}). The restriction flag is set at node d, and its backup label is effectively reinitialized. The primary label at node c is not of a higher cost than a path through node b extended to node c via bc, and so label $L_1(c)$ is not changed. In a next iteration, the label at node c is made permanent, its backup label is removed from the priority queue, and primary labels of node f and node e are changed to $L_1(f)=(5, \{acf\})$ and $L_1(e)=(4, \{ace\})$ respectively. The restriction flag is set at node e. Neither label $L_1(d)$ nor label $L_2(d)$ is changed when node c is processed because node d has a lower cost primary label than that of the path through node c, and because the path through node c is not allowably extended, the secondary path to node d cannot be updated. In a next iteration the primary label at node d is made permanent, its initialized backup label is retained, and no adjacent label is updated because the restriction flag is set at node d, and the only impermanent label of an adjacent node is at node e, which is in the same serial restriction group as node d.

[0062] A next iteration makes the label at node e permanent. The backup label at node e is retained in the priority queue. The backup label at node d is not updated because aced is not allowable and the restriction flag is set at node e, and node f is not relabeled by the update because it already has a lower cost route than an extension of the path through node c. However, the primary label of the path to node g is set to (5, {aceg}). While the subset restriction flag is set at node e, and node g is in a serial restriction group, the path is still permitted because node e and node g are in different serial restriction groups (corresponding to subnetworks 14*a* and 14*b*, respectively). The primary label of node f is the next label made permanent, resulting in node h being updated, which receives a label $L_1(h)=(7, \{acfh\})$. The backup label of node e is also updated and changed from the initialized value to (8{acfe}).

[0063] In a subsequent iteration, the primary label of node g is made permanent, its backup label is dropped, and node i and node j are relabeled as follows: $L_1(i)=(8, \{acegi\})$ $L_1(j)=(9, \{egj\})$ and restriction flags are set at both of these nodes. Node h is also updated so that the restriction flag is set, and the labels are changed as follows $L_1(h)=(6, \{aceh\})$ and $L_2(h)=(7, \{acfh\})$. Depending on whether the subset sequence constraint is the stronger constraint or subset intransitivity, label $L_2(e)$ is not updated, or is updated to (6, {acege}). In this example, the network is subject to the stronger subset sequence constraint.

[0064] At this juncture, the priority queue contains the following labels: $L_1(h)=(6, \{aceh\})$, $L_2(h)=(7, \{acfh\})$, $L_2(e)=(8, \{acfe\}$ [alternatively (6, {acege}), if merely subset intransitive], $L_1(i)=(8, \{acegi\})$, $L_1(i)=(9, \{acegj\})$ and backup labels of nodes d, i and j are initialized. In a next iteration, the primary label of node h is therefore made permanent, but no labels can be updated as all impermanently labeled nodes adjacent to node h are in the same serial restriction group as node h, and the restriction flag is set at node h. The backup label of node h is subsequently made permanent, and node h is processed again via a secondary path acfh. Subsequently, the primary label of node j is updated as follows label $L_1(i)=(8, \{acfj\})$, and the restriction flag remains set at node j because node h and node j are in the same serial restriction group. The primary label of node i is also updated but results in no change.

[0065] The backup label of node e is next to be made permanent (if the subset sequence constraint were subset intransitivity, the order in which labels $L_1(h)$, $L_2(h)$ and $L_2(e)$ are processed is different, but nothing else changes viz. primary labels). Label $L_2(d)$ cannot be updated as the path acfed (or equally aceged) is not allowably extended to a node in the serial restriction group. The primary label of node i is next made permanent: its initialized backup label is retained, and again no other label can be updated because of the restriction flag set at node i. Lastly node j is made permanent, its still initialized backup label is also retained and the priority queue contains only backup labels for nodes d, i, and j. The procedure therefore ends.

[0066] The invention has therefore been described in terms of two versions of a procedure and two examples of an application of the procedures. The procedures compute an optimal path from a root node to other nodes in a graph compensating for a subset sequence constraint, and therefore permits the identification of optimal allowable routes through a network represented by the graph.

[0067] The embodiments of the invention described above are intended to be exemplary only. The scope of the invention is therefore intended to be limited solely by the scope of the appended claims.

We claim:

1. A method for computing an optimal route between a first network element (NE) and other NEs of a network using a weighted graph of interconnected nodes that represent the network, the optimal route being subject to a subset sequence constraint associated with a serial restriction group in the weighted graph, the method comprising:

creating a list of temporary labels respectively associated with the nodes of the graph, each node being assigned a primary label, and each node in the serial restriction group being additionally associated with a backup label, the list initially comprising a most optimal primary label of a root node that represents the first NE;

examining the list of temporary labels to identify a most optimal label, and making the identified label permanent to remove it from the list;

selectively updating primary labels in the list for every node adjacent the permanently labeled node in order to ensure that the primary labels identify an optimal allowable path from the root node to the adjacent node; and

repeating the examining and selectively updating until all primary labels of nodes representing the other NEs are permanent.

2. The method as claimed in claim 1 wherein selectively updating further comprises:

selectively updating backup labels in the list for every node adjacent the permanent labeled node in order to ensure that the backup labels identify an optimal allow-

TRANSMITTAL FORM

(to be used for all correspondence after initial filing)

Application Number	10/787,107
Filing Date	February 27, 2004
First Named Inventor	Hong Zhang
Art Unit	2661
Examiner Name	-
Attorney Docket Number	9-13528-202US
Total Number of Pages in This Submission	11

ENCLOSURES (Check all that apply)

- [ ] Fee Transmittal Form
  - [ ] Fee Attached
- [ ] Amendment/Reply
  - [ ] After Final
  - [ ] Affidavits/declaration(s)
- [ ] Extension of Time Request
- [ ] Express Abandonment Request
- [ ] Information Disclosure Statement
- [ ] Certified Copy of Priority Document(s)
- [ ] Reply to Missing Parts/ Incomplete Application
  - [ ] Reply to Missing Parts under 37 CFR 1.52 or 1.53
- [ ] Drawing(s)
- [ ] Licensing-related Papers
- [ ] Petition
- [ ] Petition to Convert to a Provisional Application
- [ ] Power of Attorney, Revocation Change of Correspondence Address
- [ ] Terminal Disclaimer
- [ ] Request for Refund
- [ ] CD, Number of CD(s) ____
  - [ ] Landscape Table on CD
- [ ] After Allowance Communication to TC
- [ ] Appeal Communication to Board of Appeals and Interferences
- [ ] Appeal Communication to TC (Appeal Notice, Brief, Reply Brief)
- [ ] Proprietary Information
- [ ] Status Letter
- [✓] Other Enclosure(s) (please identify below):

Publication Correction

Remarks

SIGNATURE OF APPLICANT, ATTORNEY, OR AGENT

Firm Name	Ogilvy Renault, LLP		
Signature	[illegible]		
Printed name	Max R. Wood		
Date	October 28, 2005	Reg. No.	40,388

CERTIFICATE OF TRANSMISSION/MAILING

I hereby certify that this correspondence is being facsimile transmitted to the USPTO or deposited with the United States Postal Service with sufficient postage as first class mail in an envelope addressed to: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450 on the date shown below:

Signature	[illegible]		
Typed or printed name	Max R. Wood	Date	October 28, 2005